Sri Kankanahalli

Natalie Kelly

Independent Research

12 February 2010

Artificial Intelligence: Using Neural Networks for Image Recognition

Abstract:

The engineering goals of this experiment were to create a computer program using neural networks able to consistently and accurately (accuracy defined as recognizing above 90% of images correctly) classify and identify handwritten or typed text. The program should easily adapt to different situations, such as facial and handwriting recognition. The image recognition in this project focused on handwriting and type recognition. The neural network model used was a supervised Kohonen network. The images were pre-processed using Sobel edge detection and Hough transforms to find the main lines and features in different parts of the images, and histograms to calculate the mean and standard deviation. The final product of this project met all of its engineering goals; it consistently recognized almost all typed and most handwritten letters. The accuracy rate of the final test was 91%, with only 3 relative minor errors out of the 36-character alphanumeric set; this is in comparison to the first working test of the project, with a 52% accuracy rate.

This project has many applications; handwriting recognition is currently used in personal digital assistants (PDAs), phones, forensics, and many other areas. Image recognition in general has even more sweeping uses, in fields such as biometrics and robotics. The engineering goals for this project were to create a computer program for image recognition, specifically aimed at type and handwriting recognition.

Introduction:

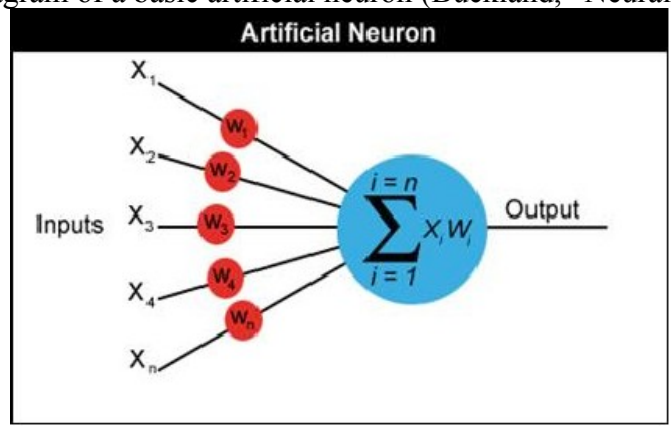
Artificial intelligence is defined by the Gale Encyclopedia of Science as being “a subfield of computer science that seeks to...[create] software and hardware that possesses some of the behavioral flexibility shown by natural intelligences, both human and animal” (Lerner 1). Essentially, the field of artificial intelligence (abbreviated as AI) is focused on creating computer programs and machines able to flexibly adapt to different situations, in a similar way to a human (or animal); it attempts this by using algorithms and defined rules to mimic human behavior. All AI systems have the ability to learn from data received over time; this learning is accomplished through a learning algorithm, which is a predefined set of rules and actions that simulate learning.

The learning algorithm used for the program constructed in this experiment was the artificial neural network. An artificial neural network is designed to simulate the structure of the human brain and the way it processes information. The human brain consists of millions of individual cells (biological neurons) connected together to form a biological neural network. Each individual biological neuron is extremely limited in how it processes information; it takes inputs and changes them in simple ways to create outputs, which it sends out. The individual neuron is only able to perform simple tasks, but when many neurons are connected or arranged together, the network can tackle much more complex problems (Buckland, “Neural Network” 2).

An artificial neural network works in a similar way; it too consists of many artificial neurons connected or arranged together to perform a task (in this experiment, that task was letter recognition). Each artificial neuron has numeric inputs, weights, and outputs. The inputs are the data fed into the neuron, and the outputs are the values the neuron sends out. The weights determine exactly how the neuron changes the input. For every input the neuron receives, it has

one corresponding numeric weight; to calculate the neuron's final output, each input is multiplied by its corresponding weight, and then added together with the other modified inputs to get the neuron's final output (Buckland, “Neural Network” 2).

Figure 1: Diagram of a basic artificial neuron (Buckland, “Neural Network” 2).



Neural networks learn by adapting their weights to the inputs they receive, and the corresponding outputs, in a mathematical way (this can differ depending on how the neurons are connected, what the programmer wants the network to accomplish, etc.), so that the network will improve in achieving its defined task over time. This process is called training the network. There are two types of training, supervised training and unsupervised training. In supervised training, the network knows what output it is supposed to get and tries to modify its weights to match, while in unsupervised learning, the network does not know what the output should be and must learn on its own. (Matthews, “An Introduction” 3) The final version of the neural network program in this experiment used a supervised learning approach.

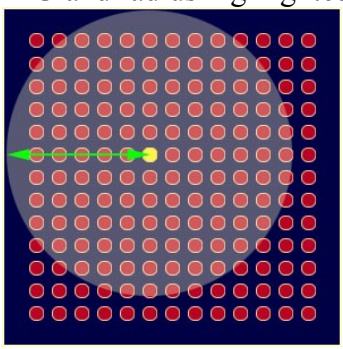
The ability of neural networks to efficiently learn and adapt to new situations is an advantage over other learning algorithms. Neural networks are used in many fields, from original music composition to robotics to (in this experiment) optical character recognition, or OCR. An article by Jeff Heaton gives a definition of OCR and an example of how it can be used: “OCR

programs are capable of reading printed text. This could be text that was scanned in from a document, or hand written text that was drawn to a hand-held device, such as a Personal Digital Assistant (PDA). OCR programs are used widely in many industries. One of the largest uses of OCR systems is the United States Postal Service. In the 1980's the US Postal Service had many Letter Sorting Machines (LSMs). These machines were manned by human clerks that would key the zip codes of sixty letters per minute. These human letter sorters have now been completely replaced by computerized letter sorting machine. This new generation of letter sorting machines is enabled with OCR technology." (Heaton 193) Essentially, OCR systems are able to take images of printed text or handwriting and recognize the letters in that image.

The type of neural network used in this experiment was the Kohonen network. Unlike most traditional neural network models, in a Kohonen network, neurons are arranged in a two-dimensional grid (usually in a rectangle). The Kohonen network is based on *competitive learning*; in a competitive learning algorithm, neurons are not all trained at once, but instead must "compete" with each other in some way in order to get trained (Matthews, "Self-Organizing Nets" 2).

A Kohonen network is trained by presenting the network with a randomly chosen set of inputs from the total input data it is being trained on (consisting of many different sets). The neuron that has the highest output out of the entire network for that input set is designated the best matching unit (BMU). The BMU, as well as its surrounding neurons within a certain radius, are then trained based on the current input set; the farther away a neuron is from the BMU on the network grid, the less it is trained. The BMU's radius also decreases over time; at a certain point, the radius will shrink until it only contain the BMU itself. At that point, training is said to be complete (Buckland, "SOM Tutorial" 3).

Figure 2: A Kohonen network. BMU and radius highlighted (Buckland, “SOM Tutorial” 3).



The output of a Kohonen network is determined by looking at which neuron was the BMU (or which group of neurons contained the BMU). For example, if the network was being used for OCR, if the first neuron (top-left) in the network was the BMU the network's output would be interpreted as the letter “A.”

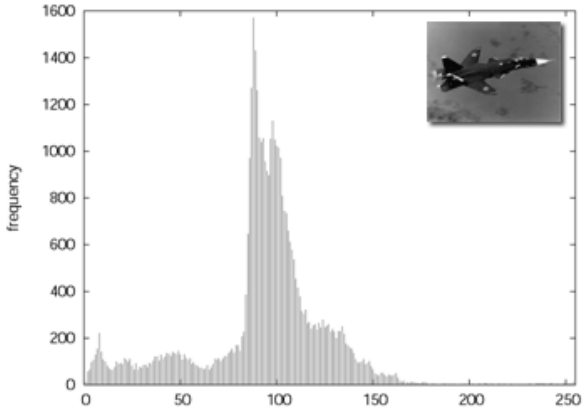
A useful aspect of the Kohonen network is its inherent classification ability. As the network is trained, “regions” begin to form in the network grid; similar inputs are clustered into the same neuron or same group of neurons (Buckland, “SOM Tutorial” 1). This attribute helps in identifying noisy and corrupt data. For example, if many photographs of different locations were shown to the network, then similar photographs would group into the same neurons (e.g. pictures of sunsets might go into neuron 1, pictures of sunrises might go into neuron 2, etc.)

However, one cannot present whole images to a neural network; this is too much information for the network to handle, and there is a lot of essentially unimportant data which may confuse the network (e.g. not being able to recognizing letters at different angles). The images must go through some form of preprocessing before being shown to the network, in which the most important features of the image (in this experiment, the letters) are extracted and presented in a numerical form. For the final Kohonen network constructed in this project, many preprocessing algorithms and techniques were applied to the letter images before presenting

them to the network. The image of the letter was first split into 25 equally sized subsegments, and then several algorithms were applied to each segment; these algorithms were the histogram, the Hough transform, and the Sobel edge detector.

A histogram is essentially a graph showing the color distribution of an image, from light to dark. When represented on a computer, each color is created by combining different intensities of three color components (red, green, and blue). A histogram maps these intensities for each one of these three colors; as defined in an article by James Matthews, "an image histogram maintains a count of the frequency for a given colour level. When graphed, a histogram can provide a good representation of the colour spread of the image." (Matthews, "A Basic Introduction" 8) Like any graph, a histogram graph has both a mean and a standard deviation. The mean and standard deviation for each color component in each image segment were fed into the Kohonen network created in this experiment as inputs.

Figure 3: A histogram graph of a greyscale image.



An algorithm called the Hough transform was also applied to each image segment. Robert Fisher and the other authors of the HIPR2 image processing library describe the Hough transform as "a technique which can be used to isolate features of a particular shape within an image. Because it requires that the desired features be specified in some parametric form, the

classical Hough transform is most commonly used for the detection of regular curves such as lines, circles, ellipses, *etc*." (1) A Hough transform is used to isolate the main lines in an image and find their parameters (the angle and y-intercept). For each of the 25 subsections of the image being fed into the Kohonen network, a Hough transform was used to find the one main line of that subsection. After this, the sine and cosine of that main line's angle, as well as its y-intercept, were given to the network.

The final algorithm used in preprocessing was Sobel edge detection. Edge detection, as defined by Bill Green's article for Drexel University, is highly important because "edges characterize boundaries and are therefore a problem of fundamental importance in image processing. Edges in images are areas with strong intensity contrasts – a jump in intensity from one pixel to the next. Edge detecting an image significantly reduces the amount of data and filters out useless information, while preserving the important structural properties in an image**.**" (4) Edge detection filters out noise in the image and preserves its basic structure; this is key when dealing with images such as letters, where the structure is the most important aspect. The Sobel edge detection algorithm calculates the derivative of the image (rate of change), and uses that to find edges; the final output is a new grayscale image with only edges showing. A Sobel edge detector is run on each of the 25 subsegments for each image, and the histogram mean and standard deviation of the output grayscale image are fed into the network as inputs.

The Kohonen network used in this experiment has a 6x6 grid of neurons, and recognizes 36 characters (26 letters, 10 numbers). Each character maps to exactly one neuron. The network uses a form of supervised learning; the network already knows which neurons are associated with particular characters (for example, the network knows that the letter A must map to neuron 1 and the letter F must map to neuron 6).

Materials and Methods:

Materials necessary for this project were a computer running Windows and a compiler for the Blitz3D programming language.

1. The resources read from various sources were used to design the structure and plan the implementation of the neural network (what type of network, how many neurons total, etc.)
2. A prototype of the neural network program was created, using the Blitz3D programming language (based off BASIC).
3. The prototype program's performance was evaluated in optical character recognition (OCR). The network learned to recognize the 26 letters of the alphabet, as well as the 10 numbers (so 36 characters total).
4. For OCR testing, the neural network was first trained on 3 sets of images of all 36 characters, in different fonts and handwriting.
5. The network was presented with a new set of images it had not seen before, and classified each image as one of the 36 characters it has learned. The number of characters classified correctly and the number of characters classified incorrectly was recorded, and the program ran 3 times for each set.
6. Based on the data collected for each set, changes were made to the prototype, and testing began again from step 3 (with the network being retrained). This entire process constituted one trial.

Results:

After being trained on three sets (including both fonts and handwriting) of characters, the network was tested on a typed font it had never seen before. On average, when tested on typed fonts, the final version of the network was able to recognize letters with a 91.6% accuracy rate (on average missing 3 out of every 36, an ~8.3% error rate). This is a vast improvement from the first working trial (trial 3), where the accuracy rate was only about 52.2%, and the error rate was an astonishingly high 47.7%. The final version of the network used a supervised learning approach and divided the letter images into 25 subsegments, using three image preprocessing techniques (histogram, Sobel edge detection, and Hough transform), while the first working version of the network used an unsupervised learning approach and only divided the letter images into 16 subsegments, using only two image preprocessing techniques (histogram and Hough transform).

Complete experimental notes and data tables are in the appendix.

Discussion:

The first working network's large percent error (in trial 3) was probably due to the unsupervised learning approach that was initially taken. Each character must be associated with exactly one neuron and one neuron only; there are 36 total neurons in the network, and 36 letters which need to be classified. However, with an unsupervised approach, the network did not know that each character must be mapped to exactly one and only one neuron; therefore, when the network used unsupervised learning, it was mapping several characters to the same neuron, or even to none at all. This meant that the network was not able to distinguish between letters well, and led to many problems; this is why a supervised learning approach had to be taken from that

point.

However, even this supervised approach is not perfect; there are many patterns in the data which show the network's faults. In the final trial of the network, the network was mainly confused between letters with similar structure (for example, it sometimes classifies the letter G as the number 6, the number 0 as the letter O, or the number 5 as the letter S). There are two possible ways this could be addressed; the first way would be to simply increase the number of subsegments that letter images are divided into, so as to break down the image even more for the network and focus it on even smaller details.

The second possible way to address this problem is slightly more complicated, and involves changing the layout of the network itself. Currently, the characters in the 6x6 Kohonen network map to neurons in this specific manner:

A	B	C	D	E	F
G	H	I	J	K	L
M	N	O	P	Q	R
S	T	U	V	W	X
Y	Z	0	1	2	3
4	5	6	7	8	9

Figure 4: Diagram showing which characters map to which neurons on the network grid.

Each character maps to each neuron in a very simple way; the first letter A maps to the first neuron from the top-left, the second letter B maps to the second neuron from the top-left, and so on. However, this structure may actually interfere with the Kohonen network's classification ability; one of the key aspects of a Kohonen network's classification is data forming into regions, where similar inputs are grouped together. However, as is shown in Figure 4, similar letters are actually not put into similar regions; the neuron for the letter O is

surrounded by the neurons for the letters H, I, J, N, P, T, U, and V. This means that if, for example, the neuron O was the BMU for one iteration of training, the letters I,H,N, and T may also be affected, even though they are not similar in structure to the letter O. This is a flaw in the network design and also a source of error, which can be improved on and fixed in the future.

Another source of error which may impact the network's ability is segmentation of the letter images (or, specifically, the number of segments the image is divided into). In early trials, the number of image segments used was 16; however, in later trials, this number was raised to 25, which seemed to increase the accuracy of the network in recognizing letters. The only difference between trial 5 and trial 6 of the network was this increase; however, in trial 5 the network only had a mean accuracy rate of 86.1% whereas in trial 6 the network's accuracy rate was 91.6%, showing that the number of image subsegments does indeed have an impact on network accuracy (though not as large as the impact of supervised learning).

The only real experimental source of error (that is, not stemming from flaws in the network model itself) was the computer's random number generation. The computer's random number generator was used to initialize the weights of the Kohonen network, so that different trials would achieve different results. The random number generator is also used in training the network; a random letter or number from each font set is chosen and presented to the network. The random number generator may have a small impact on the final results of the network, but not enough to severely throw off the final results; it is not a particularly huge source of error, but it is one nevertheless.

Further advancements in this project include correcting the flaws in the network that were addressed, in order to improve the program's image recognition capabilities, and also to refine the network in order to eliminate the minor bugs, such as confusion between similar letters.

Adapting the network to handwriting as well as font recognition is also a topic worth exploring; handwriting recognition is a more complex topic than font recognition, because of the vast ramge of handwriting styles to take into account and the amount of variance in handwriting samples, even from the same person, which may confuse the network. Another topic to look into is using a Kohonen network like the one constructed in this experiment for facial recognition; the network would probably work well with this scenario, because there is much less room for network error in facial recognition than in handwriting recognition (since shots are often from the same angle, and many key features remain the same).

Works Cited

Buckland, Mat. "Neural Network Tutorial." *AI Junkie*. Ed. Mat Buckland. N.p., 6 Nov. 2003. Web. 19 Oct. 2009. <http://www.ai-junkie.com/ann/evolved/ nnt1.html>.

---. "SOM Tutorial Part 1." *AI Junkie*. Ed. Mat Buckland. N.p., 21 Dec. 2004. Web. 2 Dec. 2009. <http://www.ai-junkie.com/ann/som/som1.html>.

Fisher, Robert, et al. "Hough Transform." *HIPR2*. N.p., 2003. Web. 10 Jan. 2010. <http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm>.

Green, Bill. "Edge Detection Tutorial." *Drexel University.* N.p., 2002. Web. 4 Jan. 2010. <http://www.pages.drexel.edu/~weg22/edge.html>.

Heaton, Jeff. *Introduction to Neural Networks with Java*. N.p.: Heaton Research, Inc., 2005. Print.

Lerner, K. Lee, and Brenda Wilmoth Lerner. "Artificial Intelligence." *Gale Encyclopedia of Science*. 4th ed. Detroit: Gale Group, [2009?]. *Student Resource Center Gold*. Web. 8 Sept. 2009. <http://www.galegroup.com>.

Matthews, James. "A Basic Introduction to Image Processing." *Generation5*. N.p., 28 Nov. 2004. Web. 21 Dec. 2009. <http://generation5.org/content/2001/ im00.asp>.

---. "Self-Organizing Nets." *Generation5*. Ed. James Matthews. N.p., 24 Oct. 2004. Web. 18 Oct. 2009. <http://generation5.org/content/1999/selforganize.asp>

---. "An Introduction to Neural Networks." *Generation5*. Ed. James Matthews. N.p., 31 Mar. 2000. Web. 30 Sept. 2009. <http://www.generation5.org/content/2000/nnintro.asp>

Appendix (Data and Experimental Notes):

Trial 1:

Network Description: Kohonen network, 6x6 neuron grid, learning rate = 0.3, unsupervised (learns to classify by itself)

Pre-Processing for Inputs: Each grayscale image has its histogram mean and standard deviation calculated. The image is also run through a Sobel edge detector to get a new grayscale image, which has its histogram mean and standard deviation calculated as well. The image is then split into 16 equal subsegments, and the above process is repeated for each. This makes for 68 inputs total.

Results: Failed in preprocessing step. The program crashed with an "array index out of bounds" error.

Trial 2:

Network Description: Same as #1

Pre-Processing for Inputs: Same as #1

Results: All images mapped to the neuron for the letter "F." Upon investigation, it turns out this was the result of not clearing an array before using it again in computations.

Trial 3:

Network Description: Same as #1

Pre-Processing for Inputs: Same as #1

Results:

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
T	D	C	D	F	F	8	M	I	I	K	L	W	M	O	P	Q	T	S	Y	V	V	V	X	Z	Z	O	I	I	8	4	5	6	7	8	9
A	B	C	T	F	F	6	N	I	I	K	L	W	M	O	P	Q	T	S	Y	V	V	V	X	Z	Z	O	I	I	3	4	0	6	7	8	9
T	B	C	T	F	F	6	N	I	I	Z	L	W	M	O	P	Q	S	5	Y	V	V	V	7	Y	Z	O	I	4	3	4	0	6	7	8	9

Amount Classified Correctly for T1: 19/36 Amount Classified Incorrectly for T1: 17/36

Total Accuracy Rate for T1: 52%

Amount Classified Correctly for T2: 21/36 Amount Classified Incorrectly for T2: 15/36

Total Accuracy Rate for T2: 58%

Amount Classified Correctly for T3: 17/36 Amount Classified Incorrectly for T3: 19/36

Total Accuracy Rate for T3: 47%

Mean Amt Classified Correctly: 19/36 Mean Amt Classified Incorrectly: 17/36

Mean Accuracy Rate: 52.22%

Trial 4:

Network Description: Same as #1, except now semisupervised (each neuron is assigned a specific character).

Pre-Processing for Inputs: Same as #1

Results:

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
A	B	C	D	E	F	8	M	I	T	K	L	W	N	O	P	Q	R	S	Y	U	V	W	X	Y	Z	O	1	2	3	4	5	6	7	8	9
A	B	C	D	E	F	8	M	I	T	K	L	W	N	O	P	Q	R	5	Y	U	V	W	X	Y	Z	O	1	2	3	4	5	6	7	8	9
A	B	C	D	E	F	8	M	I	T	K	L	W	N	O	P	Q	R	S	Y	U	V	W	X	Y	Z	O	1	2	3	4	0	6	7	8	9

Amount Classified Correctly for T1: 30/36 Amount Classified Incorrectly for T1: 6/36

Total Accuracy Rate for T1: 83.3%

Amount Classified Correctly for T2: 29/36 Amount Classified Incorrectly for T2: 7/36

Total Accuracy Rate for T2: 80.5%

Amount Classified Correctly for T3: 29/36 Amount Classified Incorrectly for T3: 7/36

Total Accuracy Rate for T3: 80.5%

Mean Amt Classified Correctly: 29.33/36 Mean Amt Classified Incorrectly: 6.66/36

Mean Accuracy Rate: 81.43%

Trial 5:

Network Description: Same as #4

Pre-Processing for Inputs: A Hough transform was added, to find the main line and its 3 parameters (sin of angle, cosine of angle, radius) for both the whole image and each subsegment. This brings the count of total input features up to 119.

Results:

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
A	B	C	D	E	F	8	M	I	J	K	L	W	N	O	P	Q	R	5	Y	U	V	W	X	Y	Z	0	1	2	3	4	5	6	7	8	9
A	B	C	D	E	F	8	H	I	J	K	L	W	N	O	P	Q	R	5	Y	U	V	W	X	Y	Z	O	1	2	3	4	5	6	7	8	9
A	B	C	D	E	F	8	M	I	J	K	L	W	N	O	P	Q	R	5	Y	U	V	W	X	Y	Z	0	1	2	3	4	5	6	7	8	9

Amount Classified Correctly for T1: 31/36 Amount Classified Incorrectly for T1: 5/36

Total Accuracy Rate for T1: 86.1%

Amount Classified Correctly for T2: 31/36 Amount Classified Incorrectly for T2: 5/36

Total Accuracy Rate for T2: 86.1%

Amount Classified Correctly for T3: 31/36 Amount Classified Incorrectly for T3: 5/36

Total Accuracy Rate for T3: 86.1%

Mean Amt Classified Correctly: 31/36 Mean Amt Classified Incorrectly: 5/36

Mean Accuracy Rate: 86.1%

Trial 6 (Final):

Network Description: Same as #4

Pre-Processing for Inputs: The number of equally divided subsegments was increased from 16 to 25. This brings the number of total inputs up to 181.

Results:

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
A	B	C	D	E	F	6	H	I	T	K	L	M	N	O	P	Q	R	5	T	U	V	W	X	Y	Z	0	1	2	3	4	5	6	7	8	9
A	B	C	D	E	F	6	H	I	J	K	L	M	N	O	P	Q	R	5	T	U	V	W	X	Y	Z	0	1	2	3	4	0	6	7	8	9
A	B	C	D	E	F	G	H	I	T	K	L	M	N	O	P	Q	R	5	T	U	V	W	X	Y	Z	O	1	2	3	4	5	6	7	8	9

Amount Classified Correctly for T1: 33/36 Amount Classified Incorrectly for T1: 3/36

Total Accuracy Rate for T1: 91.6%

Amount Classified Correctly for T2: 33/36 Amount Classified Incorrectly for T2: 3/36

Total Accuracy Rate for T2: 91.6%

Amount Classified Correctly for T3: 33/36 Amount Classified Incorrectly for T3: 3/36

Total Accuracy Rate for T3: 91.6%

Mean Amt Classified Correctly: 33/36 Mean Amt Classified Incorrectly: 3/36

Mean Accuracy Rate: 91.6%